

Lecture 11: PKI, SSL and VPN

Information Security –
Theory vs. Reality
0368-4474-01, Winter 2011

Yoav Nir

What is VPN

- VPN stands for Virtual Private Network
- A private network is pretty obvious
 - An Ethernet LAN running in **my** building
 - Wifi with access control
 - A locked door and thick walls can be access control
 - An optical fiber running between two buildings



What is a VPN

- What is a non-private network?
 - Obviously, someone else's network.
- Obvious example: the Internet.
- The Internet is pretty fast, and really cheap.
- What do I do if I have offices in several cities
 - And people working from home
 - And sales people working from who knows where
- We could use the Internet, but...
 - People could see my data
 - People could modify my messages
 - People could pretend to be other people



What is a VPN

- I could run my own cables around the world.
- I could buy an MPLS connection from a local telco
 - Bezeq would love to sell me one. They call it IPVPN, and it's pretty much a switched link just for my company
 - But I can't get that to the home or mobile users
 - And it's mediocre (but consistent) speed
 - And Bezeq can still do whatever it wants
 - And it's really expensive
 - The incremental cost of using the Internet is zero
- A virtual private network combines the relative low-cost of using the Internet with the privacy of a leased line.



About this talk

- We'll quickly go over the basics of crypto
 - Can't understand VPNs without it.
- We'll talk about digital signatures and PKI
- Protecting web sites with SSL
 - And some recent spectacular fails.
- VPNs
 - IPsec/IKE
 - GRE/L2TP
 - SSL



What is a VPN

- How do we want from a VPN solution:
 1. Even with the ability to sniff my packets on the Internet, you can't tell what I'm sending.
 2. They can't modify my packets without me knowing it.
 3. They can't send me traffic and have me think it came from one of my sites.
 4. They can drop some of my packets, but they can't drop a class of packets
 - Because of #1

- We achieve all this through cryptography:
 - Encrypting packets
 - Using a MAC function to protect them
 - Secure Key distribution

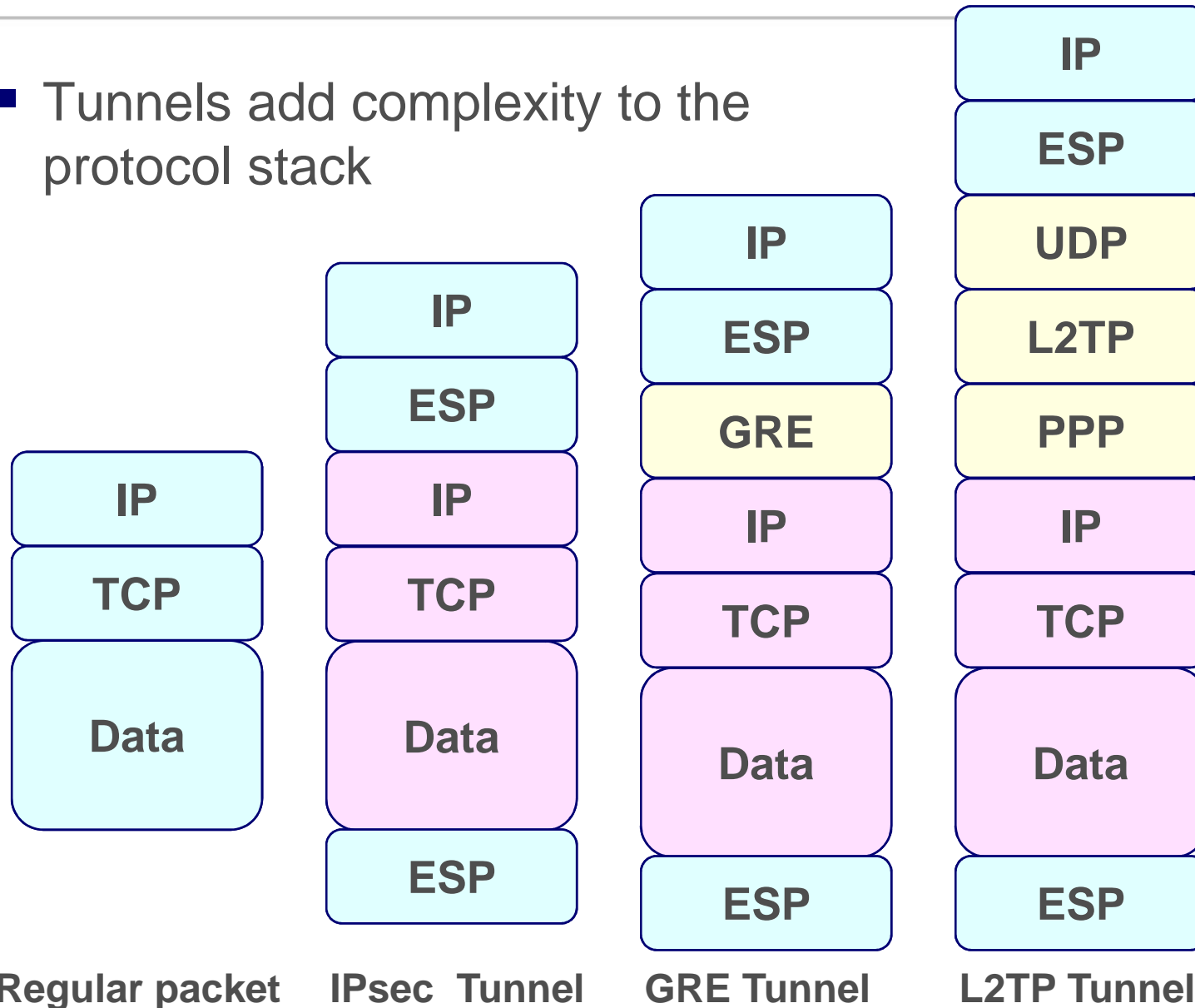


- An important concept for VPNs is a “tunnel”
- The idea is to create an “virtual overlay network” over the Internet
 - So if your company has offices in Tel Aviv and Beijing, it’s just one hop from an external router in Tel Aviv to the external router in Beijing
 - Even if on the Internet there are 20 hops going through 5 different countries.
- The packets do actually travel through all those hops, but the encryption and MAC makes it invisible to the points on the way.
 - Also, hop counts/TTLs don’t get decremented.

- Symmetric encryption + MAC protect the tunnel.
- Secure key distribution is part of setting up a tunnel.
- It's analogous to access control for the VPN.
 - You only get a key if policy says you should get a key
- Secure key distribution involves:
 - Authentication – you have to know it's actually your other office, or your employee/student calling.
 - Key distribution in such a way that an I can't tell what the key is by looking at a wireshark capture of the key exchange
- We get these things through the use of key derivation functions and the diffie-Hellman or RSA key exchange protocols.

Tunnels

- Tunnels add complexity to the protocol stack



- IPsec tunnels are one way to create the virtual overlay network. There's another way.
- You could use some other means of establishing a tunnel.
 - GRE – Generic Routing and Encapsulation adds a simple header to traffic, and has its own protocol number (47)
 - Used mainly by Cisco for site-2-site VPNs.
 - L2TP – Layer-2 Transport Protocol establishes a point-2-point protocol (PPP) link over UDP port 1701.
 - Used mainly for remote access in Windows, Mac, iOS, Adnroid
- Then you can encrypt the tunnel packets (GRE or L2TP) using transport mode IPsec.
 - Note that GRE and L2TP tunnels are network interfaces.

- IPsec tunnels have less per-packet overhead than encrypting other tunnels.
- So why not use them always?
 - IKE limitations – problems with password authentication.
 - SPDs and traffic selectors are too complicated
 - Just let the routing protocol decide what goes where
- So where does the security policy go?
 - Firewall?
 - The routing protocol?

Symmetric Encryption

- Symmetric encryption functions are used to encrypt data.
- We have two functions:
 - $F(k,m)$ - symmetric encryption function
 - $F^{-1}(k,m')$ - symmetric decryption function
- The following holds:
if
$$m' = F(k,m)$$
then
$$m = F^{-1}(k,m')$$



- Requirements from a symmetric cipher
 - Big key, so that you can't guess it by brute-force (2^n steps).
 - No weak keys - particularly no “identity” key
 - Big block, so that we don't use the same IV twice
 - ***Output should be indistinguishable from random data.***
 - More formally, if the cipher strength is n , then it takes 2^n steps to tell if this is not random data.

- Hash functions map an arbitrary-length message to a fixed-length *digest*. Typically this value is 128-512 bits in length, although database hash functions can be much smaller.
- For any hash function we would like the output to be indistinguishable from random data.
- If
$$H(m_1) = H(m_2)$$
then
$$m_1 = m_2$$
- This, of course, is not true, but we don't want a collision to happen by accident, or in an attack.

- **Preimage Resistance:**
 - Given h , we cannot calculate m such that $H(m)=h$
- **Second preimage Resistance:**
 - if we have m_1 , we cannot find m_2 such that:
 - » $m_2 \neq m_1$
 - » $H(m_1)=H(m_2)$
- ***Collision Resistance:***
 - We should not be able to make m_1, m_2 such that $H(m_1)=H(m_2)$

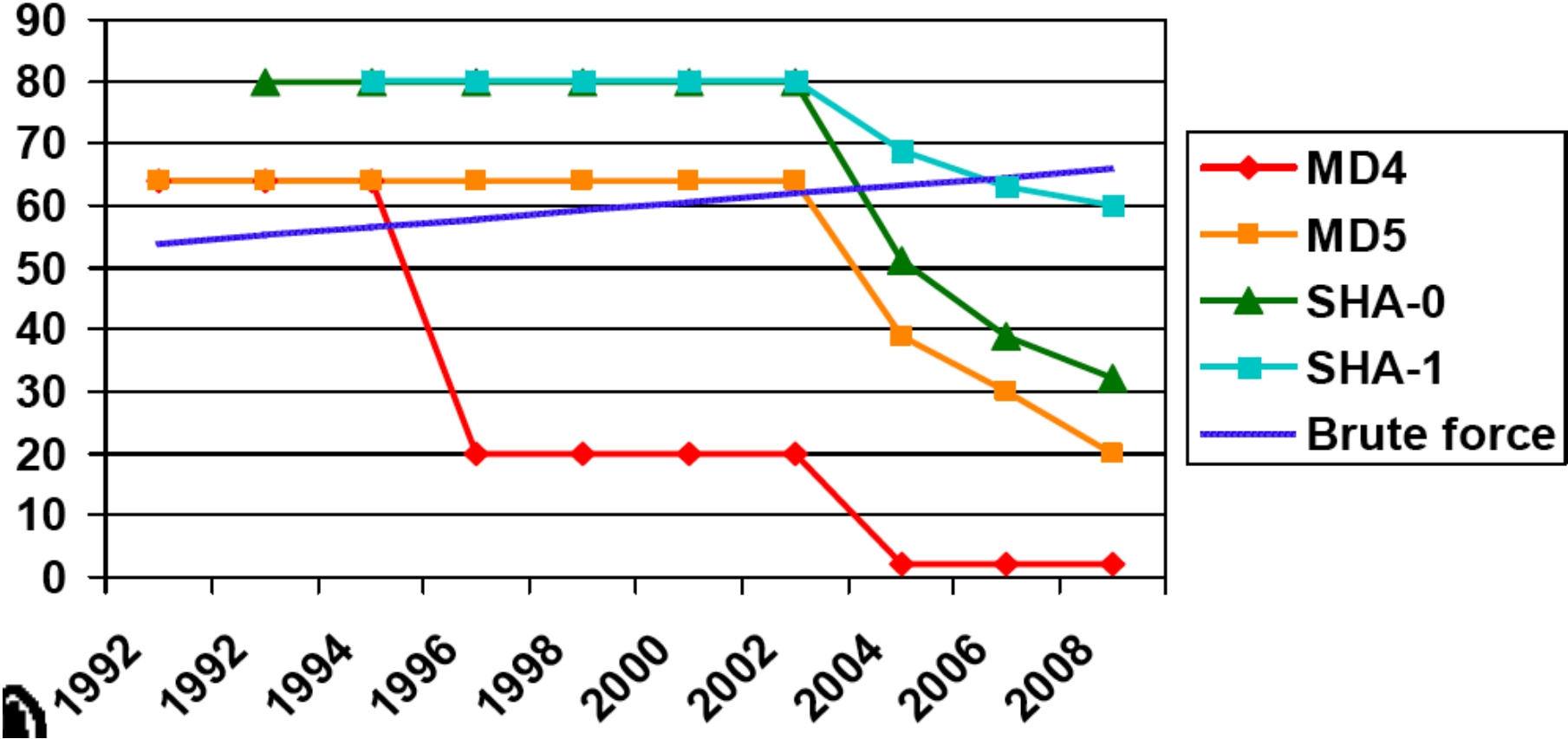
- The number of bits is important, because of the birthday attack. With n bits, you should expect one collision after $\sqrt{2^n} = 2^{n/2}$ digests.
- MD5
 - 128 bits
- SHA-1
 - 160 bits
- SHA-256
 - 256 bits

- Suppose brute-force is the best attack
- $n=128$
 - \$100,000 worth of equipment
 - 10 days
- $n=160$
 - \$500,000,000
 - 4 months
- You really need 256 bits for long term security
- Brute-force is not the best...



Hash Functions

Brute force: 1 million PCs or US\$ 100 000 hardware



Keyed Hash Function (MAC)

- Similar to a hash function, except that it takes an extra key parameter.
- You need to know the key to hash or verify the message.
- The key is secret, just like an encryption key.
- The simplest keyed hash:
 - Append the key to the message
 - Hash the result.
- HMAC is better.
- Another way is XCBC – take a block cipher in CBC mode, and use the last cipher block as the keyed hash.

Pseudo Random Function

- This is a function that creates a random-looking output from an input Seed.
- HMAC functions are the most popular PRFs.
- Sometimes called Key Derivation Functions
- They are used by protocols to take some seed, and derive all the keys and secrets
 - Encryption keys
 - MAC keys
 - Channel Bindings

- This is somewhat like a PRF, but differs in use.
 - PRFs are used in security protocols to derive keys from secret values.
 - PRNG is long lived, accepts periodic injection of entropy, and can be used continuously to derive random bits.
- Obviously: should be indistinguishable from random data.

Asymmetric Encryption

- Here we have two keys, call them k and k' , and two functions, one for encryption (E) and one for decryption (D).
- k and k' are generated together.
- If
$$m' = E(k, m)$$
then
$$m = D(k', m')$$
- For RSA, the converse is also true.

Asymmetric Encryption

1. **Generate a random 128-bit encryption key.**
2. **Use it to encrypt your mail with AES.**
3. **Encrypt the key with the RSA public key of the receiver.**
4. **Send the result along with the encrypted mail.**
5. **Receiver decrypts with their private key, and gets the AES key.**
6. **Receiver can decrypt the message.**

- This is a method to arrive at a shared secret between two parties.
- Alice wants to communicate with Bob.
 - At this point she doesn't need to know that it's really Bob at the other end.
 - Bob needs not know that it's Alice at the other end.
 - They just need a secret key so that others cannot eavesdrop on their conversation.
 - They would like to set up the secret key without having any pre-existing secret key.

- We have a universally known large prime number called p , and a generator for the multiplicative group of p , which we call g .
 - Let's suppose $p = 47, g = 2$.
- Both Alice and Bob generate random numbers smaller than p , which we will call a and b respectively.
 - $a = 22$
 - $b = 14$

- Alice sends Bob $P_a = g^a \text{ mod } p$.
 - $P_a = 2^{22} \text{ mod } 47 = 4,194,304 \text{ mod } 47 = 24$
- Bob sends Alice $P_b = g^b \text{ mod } p$.
 - $P_b = 2^{14} \text{ mod } 47 = 16384 \text{ mod } 47 = 28$
- Alice computes $g^{ab} \text{ mod } p = P_b^a \text{ mod } p$.
 - $k = 28^{22} \text{ mod } 47 = 42$
- Bob computes $g^{ab} \text{ mod } p = P_a^b \text{ mod } p$.
 - $k = 24^{14} \text{ mod } 47 = 42$

Diffie - Hellman

- Eve is an eavesdropper. Eve listens to all communications.
- Eve knows:
 - $g = 2$
 - $p = 47$
 - $P_a = 24$
 - $P_b = 28$
- If Diffie and Hellman are correct, Eve cannot guess that $k = 42$.



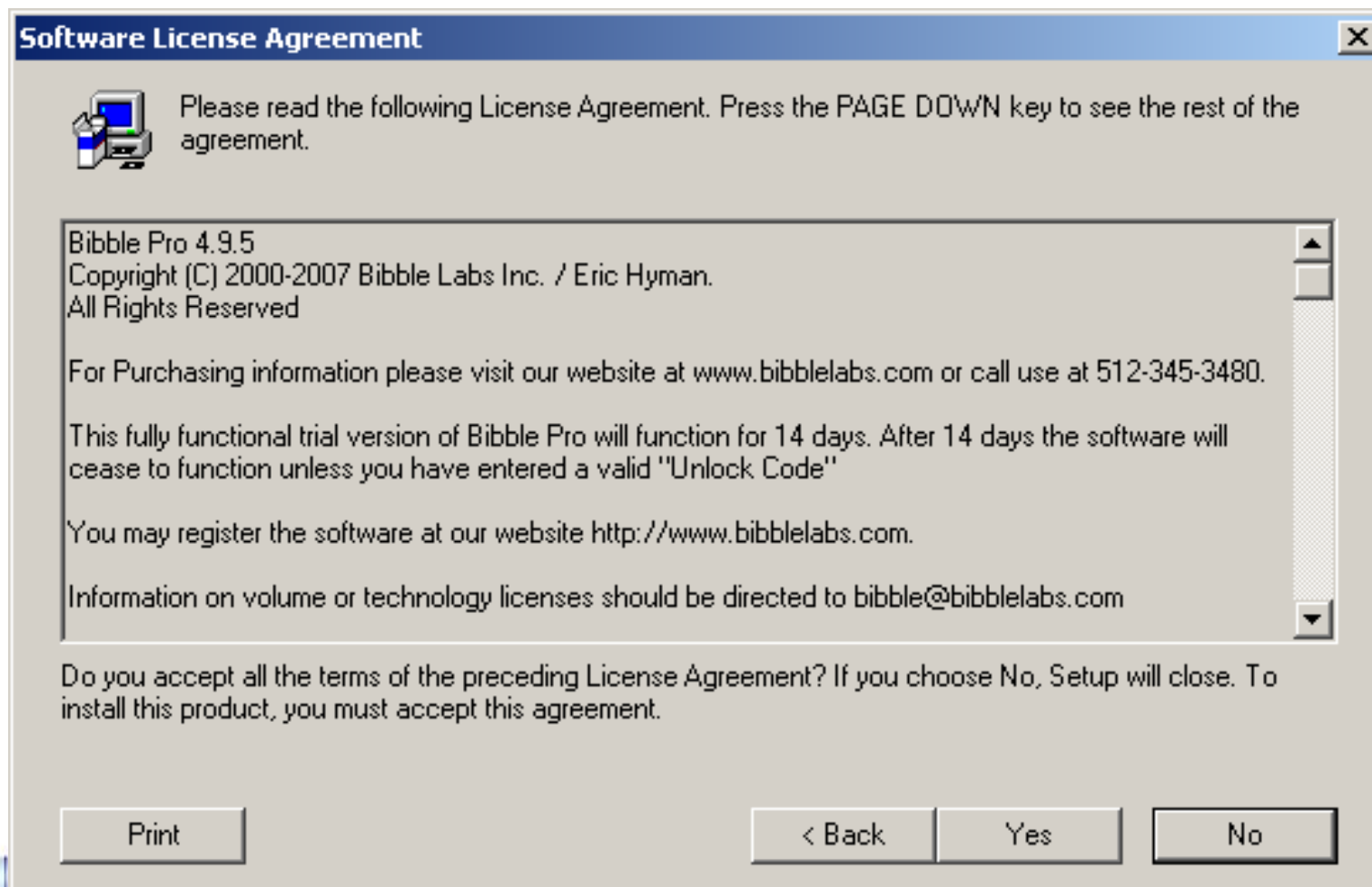
- With digital signatures one party (call her Alice) signs a message. Another party (call him Bob) can verify that Alice really signed the message.
- Requirements:
 - Non-Repudiation - Alice should not be able to claim that she did not sign the message.
 - Bob should be able to tell who signed the message.
 - Mallory should not be able to sign a message so that Bob would think it was from Alice.

- First, a word of warning:

**Electronic
Signatures
≠
Digital Signatures**

Digital Signatures

- Electronic signatures are a legal issue.



Digital Signatures - RSA

- RSA signatures rely on asymmetric encryption.
- Each party has a public key and a private key.
- We assume that only Alice knows her private key.
- We assume that Bob (and everyone else) either knows Alice's public key, or else can look it up.
- We assume that the lookup process is not vulnerable to fraud. This is the pre-existing trust for RSA signatures.

Digital Signatures – RSA/DSA/ECDSA

- Signing
 - Take the message and hash it.
$$h=H(m)$$
 - Pad the hash to the size of the keys.
 - ~~Encrypt~~ Sign the hash with the private key.
- Verification
 - Take the message and hash it.
 - Decrypt the signature using the public key.
 - Compare the hash to the decrypted value.

RSA Keying

- Used is SSL.
- Uses asymmetric encryption. Only the server needs to have a private/public pair.
- Protocol:
 - The client generates a “premaster key”
 - The client encrypts the premaster key with the server's public key.
 - The server decrypts the premaster key using its private key.
- Anyone can encrypt the master key, but only the server can decrypt it.

- When discussing RSA signatures, and asymmetric encryption, we've made some big assumptions:
 - Only Alice knows her private key.
 - Everyone can lookup Alice's public key.
 - Alice doesn't lose control of her private key, or if she does, Bob will not trust any documents signed after the fact.
- How can we get all these things?

- To get trust between two parties, we need a neutral and trusted third party.

- Think of

- I have

- government

- It states

- It has

- It is o

- Everyo

government says so!

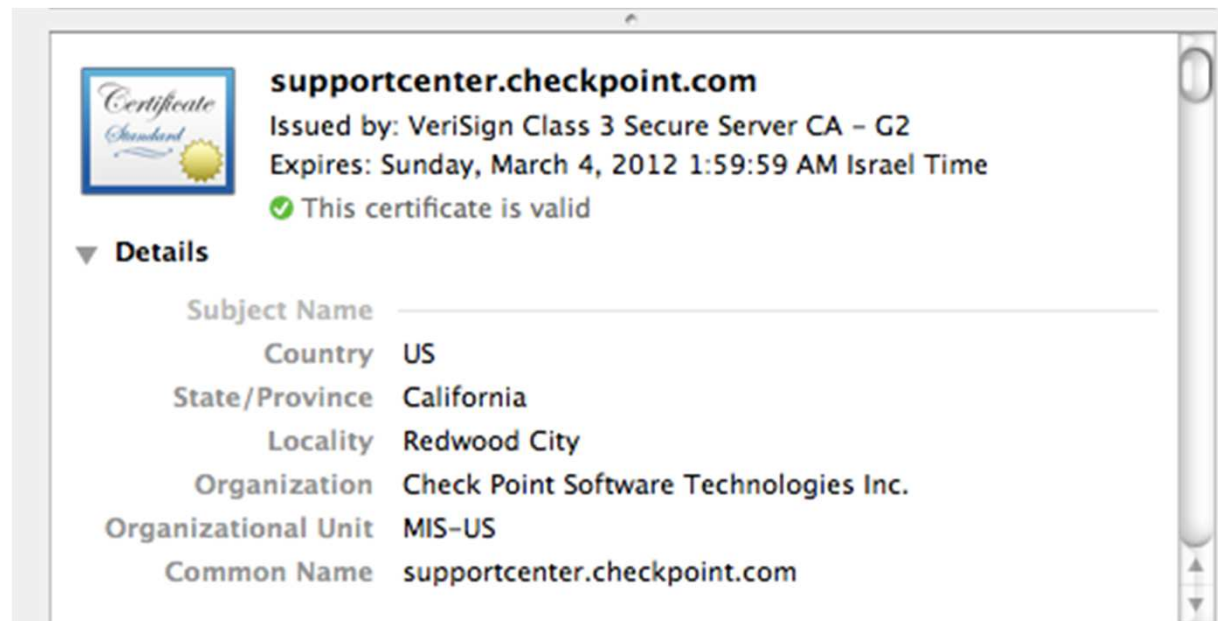


- In PKI the trusted third-party is called a CA, or certification authority.
- Everyone trusts this CA, and everyone knows the CA's public key.
- The CA signs messages called certificates. These include:
 - A subject name.
 - A serial number.
 - The subject's public key.
 - Validation criteria.

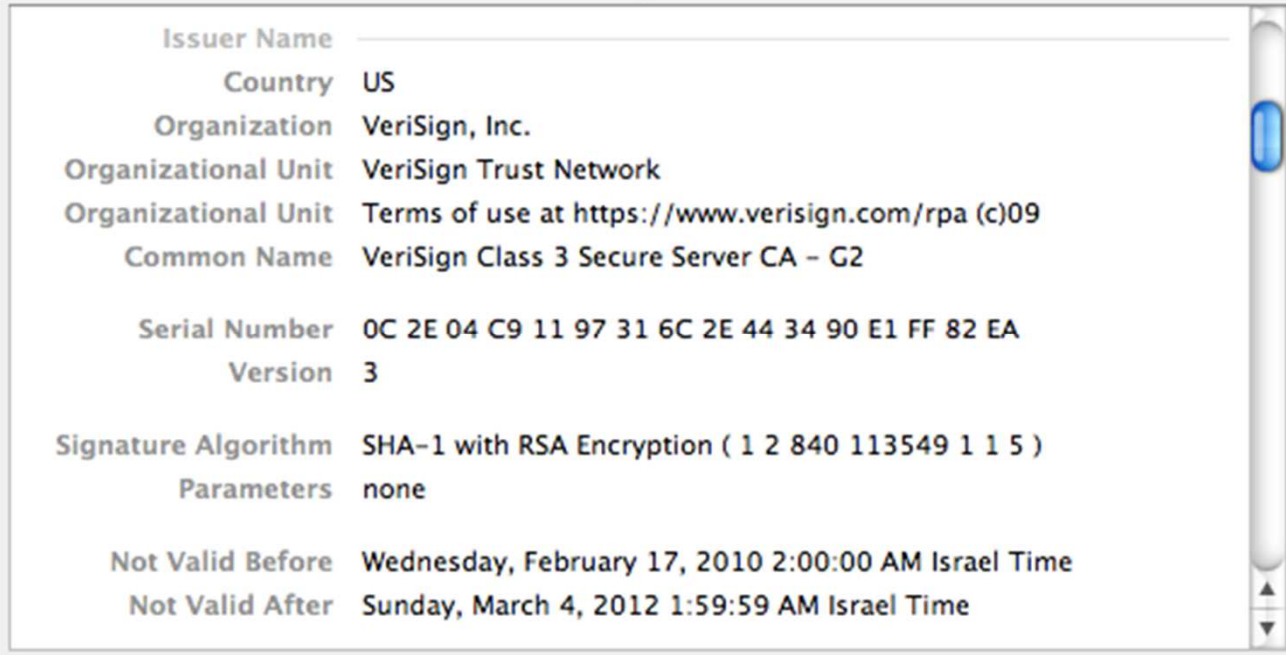
- Now Bob doesn't need to look up Alice's public key. Alice sends him her certificate.
- Bob verifies that the certificate is actually signed by the CA. If the signature verifies, Bob knows that everything in the certificate is true.
- Now Bob knows that the public key in the certificate actually belongs to Alice.
- Bob verifies the signature on the message using the public key in the cert.

- So how does it work? How does Alice get a certificate?
 - Alice creates a Certificate Request (usually in the PKCS#10 format) and submits it to the CA
 - The CA verifies Alice's identity (how?), verifies that the certificate is acceptable, and signs it.
 - Alternatively, the CA can generate the certificate and Alice can receive it by some OOB method.
- Question: What does the verification include?

- Let's take a look at a certificate. First thing to see is the subject.
- **DN is** CN=supportcenter.checkpoint.com,OU=MIS-US,O=Check Point Software Technologies Inc., L=Redwood City,ST=California,C=US



- Who issued this certificate? How is it signed? When is it valid?
- Question: Why do we need a serial number?



Issuer Name	
Country	US
Organization	VeriSign, Inc.
Organizational Unit	VeriSign Trust Network
Organizational Unit	Terms of use at https://www.verisign.com/rpa (c)09
Common Name	VeriSign Class 3 Secure Server CA - G2
Serial Number	0C 2E 04 C9 11 97 31 6C 2E 44 34 90 E1 FF 82 EA
Version	3
Signature Algorithm	SHA-1 with RSA Encryption (1 2 840 113549 1 1 5)
Parameters	none
Not Valid Before	Wednesday, February 17, 2010 2:00:00 AM Israel Time
Not Valid After	Sunday, March 4, 2012 1:59:59 AM Israel Time

Public Key Info

Algorithm RSA Encryption (1 2 840 113549 1 1 1)

Parameters none

Public Key 128 bytes : CB F7 2F 90 7F 91 FD 1A E4 CA 72 16 EE
EA A8 89 A9 60 31 00 1F 8A CA AE 58 B8 60 E7 DC 41
C3 98 5C 46 9A 64 0C DD 45 15 0D 21 A5 DF CC 92
C1 DB EB 2C BF 75 6E FF 6A 0F DA 2E 48 B9 68 63 23
30 A4 61 EC BB 70 AE 12 59 D9 2C FB 5B 1A 34 10 82
6D 9A 98 A8 C0 55 0F 1B A5 9F 45 2F 54 5B C5 F7 3B
21 63 59 34 92 79 33 1A 0A 83 47 ED 1C CA FC 44 41
C6 C8 1E 70 2F E4 B5 B5 86 AD 1F 98 A4 F7

Exponent 65537

Key Size 1024 bits

Key Usage Encrypt, Verify, Wrap, Derive

Signature 256 bytes : 87 87 37 A0 A3 3F 9E F6 B8 B0 98 0B 8A
C7 7B 39 2F 0F 75 ED 80 F1 B0 47 79 EC 66 B0 98 7C
BB 57 E1 BE FA 05 08 85 E2 DA F6 7A 94 78 D7 74 76
42 FF 12 94 24 36 35 19 E2 2A 07 EF 24 2F FF 1B D8
01 44 37 AB B3 C0 5D A8 89 B2 03 59 25 28 42 CD
AC 89 8A C5 FE CD 0C F4 0F 09 54 E1 66 5E 79 FC 4A
F8 F4 D3 15 7D 99 B5 B1 66 39 DD D5 04 38 54 12
DD BB 60 DD A0 77 9A 6A 6A 78 2D 92 8C FE 01 F5
56 5C 12 62 DF B4 A9 9F 17 04 FF A5 67 BE 6F 04 77
17 82 BD 08 1D 30 67 BE 9B B2 F0 8D B0 4C 1E 19 C6
13 36 2A C9 C4 25 FE 14 02 4A 79 42 B6 DC CA DE 4E
63 E4 4D 5C 0C EC 1C F8 AA 98 9E A1 04 F6 E7 0F 33
6F 0D A3 AB 75 CC AC B4 32 18 20 D3 19 90 DF 01
77 D2 1A 1F FA BE F2 FA 2E 99 7B 21 20 BE E4 47 0C
60 D9 42 F4 04 97 36 50 5E 62 9A DD 22 F4 43 54 EC
B8 F1 A9 50 15 87 54 34 7F

- Public Key Info
 - Algorithm
 - Key
 - Key length
 - Key usage
- Signature



Critical	NO
Usage	Digital Signature, Key Encipherment
Extension	Basic Constraints (2 5 29 19)
Critical	NO
Certificate Authority	NO
Extension	Extended Key Usage (2 5 29 37)
Critical	NO
Purpose #1	Server Authentication (1 3 6 1 5 5 7 3 1)
Purpose #2	Client Authentication (1 3 6 1 5 5 7 3 2)
Extension	Authority Key Identifier (2 5 29 35)
Critical	NO
Key ID	A5 EF 0B 11 CE C0 41 03 A3 4A 65 90 48 B2 1C E0 57 2D 7D 47
Extension	Certificate Policies (2 5 29 32)
Critical	NO
Policy ID #1	(2 16 840 1 113733 1 7 23 3)
Qualifier ID #1	Certification Practice Statement (1 3 6 1 5 5 7 2 1)
CPS URI	https://www.verisign.com/rpa



Extension CRL Distribution Points (2 5 29 31)

Critical NO

URI <http://SVRSecure-G2-crl.verisign.com/SVRSecureG2.crl>

Extension (1 3 6 1 5 5 7 1 12)

Critical NO

Data 30 60 A1 5E A0 5C 30 5A 30 58 30 56 16 09 69 6D 61 67 65 2F 67 69 66 30 21
30 1F 30 07 06 05 2B 0E 03 02 1A 04 14 4B 6B B9 28 96 06 0C BB D0 52 38 9B
29 AC 4B 07 8B 21 05 18 30 26 16 24 68 74 74 70 3A 2F 2F 6C 6F 67 6F 2E 76
65 72 69 73 69 67 6E 2E 63 6F 6D 2F 76 73 6C 6F 67 6F 31 2E 67 69 66

Extension Certificate Authority Information Access (1 3 6 1 5 5 7 1 1)

Critical NO

Method #1 Online Certificate Status Protocol (1 3 6 1 5 5 7 48 1)

URI <http://ocsp.verisign.com>

Method #2 CA Issuers (1 3 6 1 5 5 7 48 2)

URI <http://SVRSecure-G2-aia.verisign.com/SVRSecureG2.cer>

Fingerprints

SHA1 08 EE 19 A9 5A 21 F9 B4 5F 8C D8 73 55 8A 31 E1 B1 D8 54 6F

MDS 73 5D AF 47 32 6F 02 1D 32 D3 7B AA 36 93 5B DF

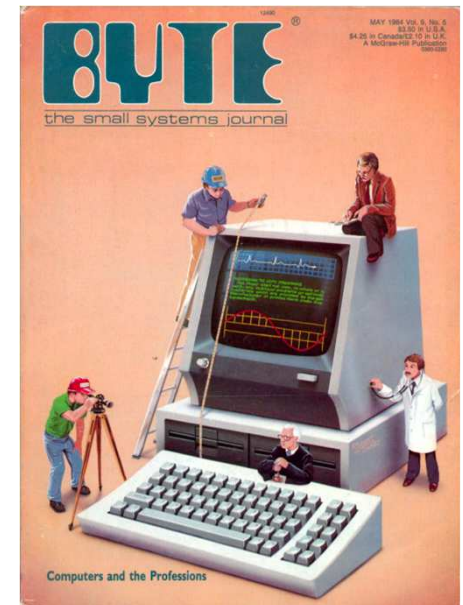


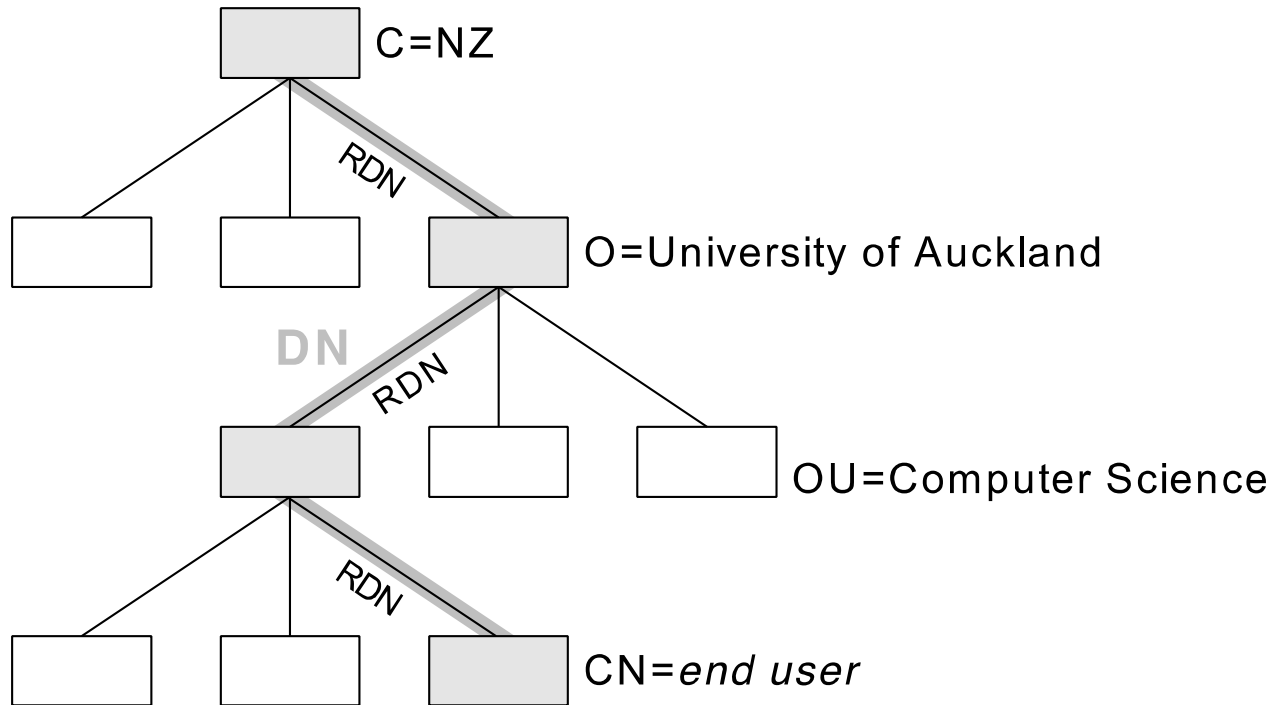
- What if Alice's private key gets stolen?
- The CA also publishes a “Certificate Revocation List”. This is a list of the serial numbers of the certificates that are revoked.
- The certificate itself contains information on where to get the CRL. That's the CDP.
- When receiving a certificate you should
 - Verify the signature (all the chain if necessary)
 - Check the CRL. For performance, cache it.

- Why is X.509 so confusing?
- All we really need is a signed structure that holds:
 - A locally meaningful name
 - A public key
- And we should be able to download this straight from the CA.
- That way, revocation is easy – take the cert off the download site.
- The reason, is that X.509 is based on X.500 directories.

PKI – Some History

- PKI was born in the late '70s
 - No “Internet”
 - No reliable communications
 - Big Telcos are monopolies (AT&T, BT, משרד הדואר)
- X500
 - Global directory run by monopoly telcos
 - Hierarchical (based on organization)
 - Path defined by a series of RDNs (relative distinguished names)
 - Collect the RDNs and you get a DN
 - The data is at the leaf of the tree (DN is also a locator)





Search key is C=NZ, O=University of Auckland, OU = Computer Science, CN = foo

- Complex way of saying `SELECT data WHERE key = 'foo'`

- Big problem with that: privacy.
 - Why should the telco expose my internal structure?
 - Can we search the leaves for single women?
 - Can we search the leaves for teenage children?

- Solution: access control
 - Each X.500 internal node has a “CA”
 - Each node has a certificate
 - No “basic constraints” – position in tree says if it’s a CA
 - No key usage – only for directory authentication
 - Just public key, validity period, and issuer+subject DNs

- There aren’t any, nor have there ever been X.500 directories. They’re not coming any time soon...

- We are not leaves. DNs don't correspond to real identities
- In the 70s, if you wanted to pay by credit card, the cashier would check the number in a little black book the the CC companies distributed.
 - This is a CRL. Hasn't been used for credit cards in 30 years.
 - Not issued frequently enough to be effective
 - Costly to distribute
 - Vulnerable to DOS attacks
 - Retroactive: now I get a CRL that says cert was revoked last week.
 - Kills non-repudiation
- Can we revoke the root CA?

- The dream of a global directory never came true.
- But locally, it can.
 - A company can give its employees certificates for authentication to the VPN
 - A government can issue its citizens identity cards with embedded certificates (Belgium does that)
 - Can you use it for access control at a Belgian company?
 - No, because there's this one French guy...
 - An industry group may create a CA and issue certificates for VPN tunnels among its members
 - But most prefer shared secrets
 - And then there's HTTPS
- All these stuff the DN with whatever stuff.

- HTTPS is a special case.
 - It's HTTP protected with SSL
- The server presents a certificate.
 - The FQDN is in the CN field or the alternate name field.
- Who's the CA?
 - Microsoft, Mozilla, Apple, Google, or Opera decide!
- They run programs to certify root CAs.
 - There are no authorizations or access control.
 - Any CA can issue a certificate for any site or sub-CA.
 - Let's see a typical list
 - Who are these, and why do I trust them?
 - There are also sub-CAs and RAs

Recent CA failures

- RapidSSL issues a rogue CA certificate to researchers
 - MD5 is only a small part of the story
- InstantSSL.it gets Comodo to issue 9 bad certificates to “ComodoHacker”
 - No cryptography involved at all
- DigiNotar issues over 500 bad certificates to “ComodoHacker”
 - No cryptography this time either
 - DigiNotar tried to keep it secret
 - Bad certificates were actually used
 - DigiNotar is out of business.

What is TLS?



What is TLS?

- TLS means Transport Layer Security.
- It was formerly known as the Secure Sockets Layer.
- It protects a layer-4 application (such as HTTP) by adding encryption and authentication to a layer-3 protocol (such as TCP or SCTP)
- TLS provides key exchange, authentication, encryption and MAC.

What is TLS?

- TLS is supported by all general-purpose computing platforms
 - SChannel – in Windows (for Explorer, IIS and SNX)
 - OpenSSL – for Apache, Safari, SNX for Mac/Linux and others. Also Check Point products
 - NSS – for Firefox
 - GnuTLS
 - Opera also has its own
 - CPTLS – in the Check Point products.
- Regular TLS uses PKI for authentication, although extensions have been defined for shared secrets.
- A TLS state can be used for one connection, although re-use is possible.

- SSL was defined by Netscape corporation in 1994.
- SSL v2 was defined by Netscape in late 1994.
- SSL v3 was released in 1995
- TLS WG formed in 1996.
- RFC published in 1999
- TLS 1.1 – 2006.
- TLS 1.2 – 2008

What is TLS

- TLS uses records with a 5-byte header and a length of up to 16K.
- There are 4 types of records:
 - Handshake
 - ChangeCipherSpec
 - Application Data
 - Alert
- Header structure:



Protocol Description - Basic

- Client sends a Client Hello:
 - Version, 32 random bytes, Session ID, Cipher-Suites, Compression Methods.
- Server sends a Server Hello:
 - Version, 32 random bytes, Session ID, Cipher-Suite, Compression Method.
- Server sends ServerKeyExchange (public key) and ServerHelloDone.
- Client sends ClientKeyExchange (encrypted secret), ChangeCipherSpec and Finished.
- Server sends ChangeCipherSpec and Finished.

- Client sends a Client Hello:
 - Version, 32 random bytes, Session ID, Cipher-Suites, Compression Methods.
- Server sends a Server Hello:
 - Version, 32 random bytes, Session ID, Cipher-Suite, Compression Method.
- Server sends **Certificate**, ServerKeyExchange (**signed this time**), and ServerHelloDone.
- Client sends ClientKeyExchange (encrypted secret), ChangeCipherSpec and Finished.
- Server sends ChangeCipherSpec and Finished.

- Client sends a Client Hello:
 - Version, 32 random bytes, Session ID, Cipher-Suites, Compression Methods.
- Server sends a Server Hello:
 - Version, 32 random bytes, Session ID, Cipher-Suite, Compression Method.
- Server sends Certificate, **CertReq**, ServerKeyExchange and ServerHelloDone.
- Client sends **Certificate**, ClientKeyExchange, **CertVerify**, ChangeCipherSpec and Finished.
- Server sends ChangeCipherSpec and Finished.

Protocol Description - Resuming

- Client sends a Client Hello:
 - Version, 32 random bytes, **Session ID**, Cipher-Suites, Compression Methods.
- Server sends a Server Hello:
 - Version, 32 random bytes, Session ID, Cipher-Suite, Compression Method.
- Server sends ChangeCipherSpec and Finished.
- Client sends ChangeCipherSpec and Finished.

- Client sends a Client Hello:
 - Version, 32 random bytes, Session ID, Cipher-Suites, Compression Methods.
- Server sends a Server Hello:
 - Version, 32 random bytes, Session ID, Cipher-Suite, Compression Method.
- Server sends Certificate, ServerKeyExchange and ServerHelloDone.
- Client sends ClientKeyExchange, ChangeCipherSpec and Finished.
- Server sends ChangeCipherSpec and Finished.
- **Can you spot what's wrong?**

Uses today and beyond

- The most common use of TLS is to protect HTTP. HTTPS is identical to HTTP.
- TLS is used to secure other protocols such as POP3, SMTP, LDAP, etc.
- TLS is used as an authentication and protection mechanism in EAP.
- TLS is used as a tunnel in SSL-VPN Products.

What is IPSec?



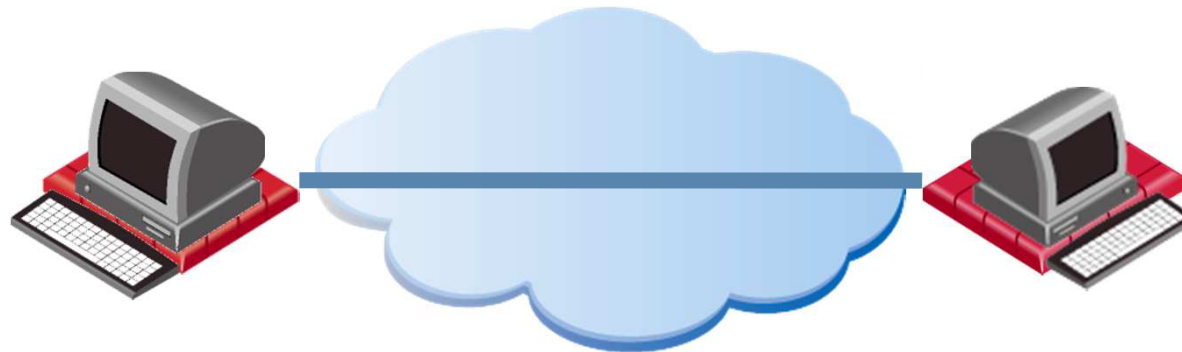
What is IPsec

- IPsec is a security protocol for the Internet.
- Allows encryption and/or authentication of IP packets.
- Works in layer 3.
 - Applications need not be aware.
- Implemented in all major OS platforms
 - Windows
 - Linux (3 options: XFRM, PFKEYv2 and KAME)
 - Mac OS X, iOS, Android



What Is a Tunnel ?

- Gateway to Gateway tunnel
- **Endpoint to Gateway tunnel**
- **Endpoint to Endpoint tunnel**

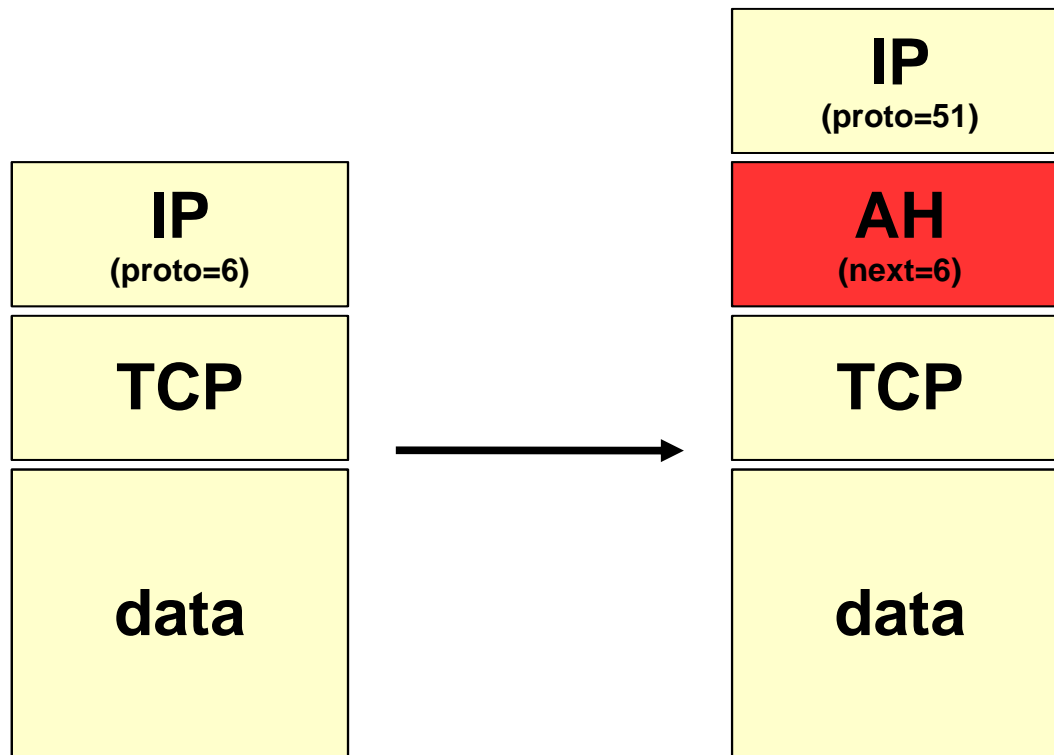


- AH - Authentication Header
 - Provides packet authentication using a keyed MAC function
 - Ensures that the packet actually came from the peer with which you exchanged keys, was not modified, and was not replayed.
- ESP - Encapsulated Security Protocol
 - Provides packet authentication *and* encryption.
 - Uses a keyed MAC and an encryption function
 - Ensures your traffic cannot be read en route.

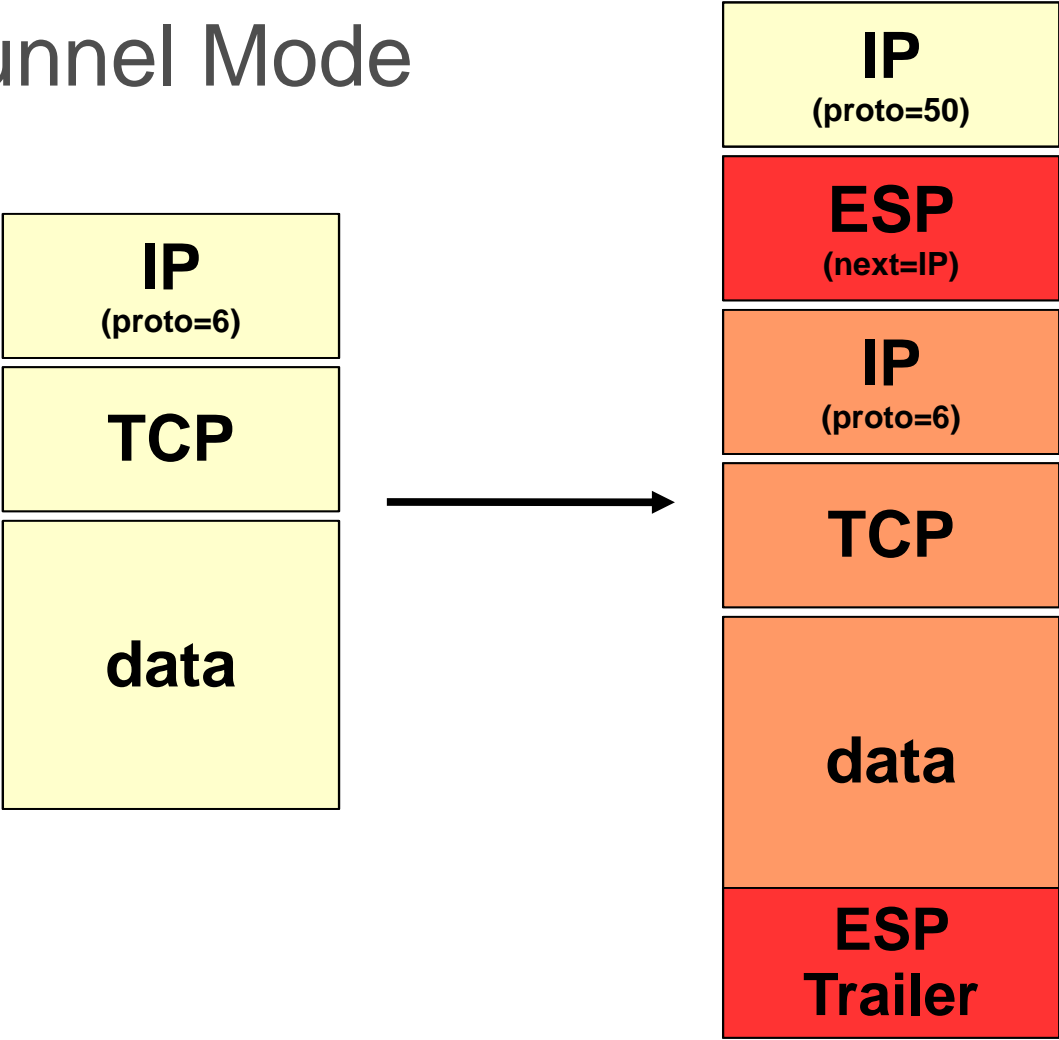
- Transport mode
 - Protects layers 4-7
 - Keeps the original IP header
 - Use it to protect a connection between peers.
 - Does not make sense for a perimeter gateway.

- Tunnel mode
 - Protects layers 3-7
 - Adds an encapsulating IP header
 - Protects networks behind the peers.

- AH in Transport Mode



- ESP in Tunnel Mode



- **Peer**

This is a machine that can do IPsec

- **Traffic Selector**

This describes one side of IP traffic. A traffic selector is defined by a list of elements as follows:

- Range of IP addresses
- IP protocol (TCP, UDP, ICMP, SCTP)
- Range of ports (if relevant)

- Using a source and destination TS, you can describe any subset of IP traffic.



■ Action

This describes what the IPsec implementation is going to do with certain traffic. Options are:

- BYPASS - let the traffic go
- DROP - make the packets disappear
- PROTECT - use IPsec on the packet

■ Methods

This says how to protect the packets: ESP? AH? Tunnel? Transport? Which algorithms? Use compression? Use TFC?



- SA = Security Association

A security association describes what to do with certain traffic. It has the following fields:

- Source traffic selector
- Destination traffic selector
- Methods & Keys
- Peer (although not explicitly in RFC 4301)
- Direction: inbound or outbound
- Time and/or volume limited.
- Has a semi-unique 32-bit value called SPI



- Security Policy (SP) element
 - Source traffic selector
 - Destination traffic selector
 - Request Flags
 - Action
 - (in real life implementations - also methods)
 - Similar to an SA, it's unidirectional, but real life implementations pair an inbound SP element with an outbound SP element, and enforce similar action and methods

- Security Policy Database (SPD)
 - List of SP elements
 - Does not change - it's a policy
 - The existence of a SP element ***does not*** imply the existence of SAs.

- SPD Cache
 - Similar in structure to the SPD
 - The existence of a SP elements ***does imply the existence of a matching SA***
 - ***Entries do not necessarily match the SPD***

- **SAD - Security Association Database**
 - Holds all the current SAs.
 - Inbound table keyed by SPI
 - Outbound table keyed by traffic selectors
 - Every entry matches an entry in SPD cache
- **PAD - Peer Authentication Database**
 - Describes the peers, and how they identify themselves (IP addresses, DNS names)
 - How peers authenticate (IKE, Kink, certs, PSK)

- Assume that the SPD covers all possible traffic.
 - That's equivalent to adding a bucket “drop” or bucket “bypass”
- There is something that creates entries in the SPD cache and the SAD. Call this thing a “daemon”
- The IPsec stack has a way of requesting SAs from this daemon

- Outbound
 - Outbound packet is matched against the SPD cache
 - If a match is found, we act on the matching SA
 - If no match is found, look in the SPD
 - If the action is BYPASS or DROP, do it, and copy to SPD cache
 - If the action is PROTECT, request an SA from the daemon, according to PFP flags
 - The daemon puts entries in the SAD and SPD cache

■ Inbound

- Packet comes in. Get the SPI from the header, and search the SAD.
- If not found, drop. Otherwise, decrypt and check the authentication.
- Check that the decrypted packet matches the SA parameters. If not, drop
- Pass the decrypted packet.
- Q: in tunnel mode, do we need to verify that the packet source is the correct peer IP address?

- Around 8 years ago, SSL VPNs were really hyped.
- Big problems with IPsec – firewalls block them.
- In its simplest form, an SSL VPN is a portal that gives the user access to company resources through a web interface:
 - email
 - Intranet
 - Fileshares, anything
- This has some advantages over IPsec tunnels:
 - Clientless – all you need is a browser
 - Web-based authentication
 - It's HTTPS and most firewalls allow HTTPS

- Soon, most vendors added some kind of tunneling client
- This would take the packets to be encrypted and send them all through a single SSL connection to port 443 – the HTTPS port.
- This gives you all the abilities of IPsec VPNs along with the firewall traversal abilities of SSL VPNs.
 - At considerably downgraded performance
 - And messing up TCP's retransmission mechanisms.
 - And ever-increasing delays in VoIP and video
- But do you think firewall makers would sit idly by while you try to tunnel all kinds of protocols through the firewall by disguising them as HTTPS?

SSL Inspection

- There is usually no good reason for anyone behind a corporate firewall to open an IPsec tunnel to some other place
 - So corporate firewalls block IPsec.
- There are good reasons to use HTTPS
 - Gmail, buying stuff on the Internet
 - So corporate firewalls don't block HTTPS
- But are we going to let just any traffic go out?
- And if a firewall is protecting HTTP traffic from various attacks (such as files infected by malware), do we allow the malware through because it came over HTTPS?
- No way.

- In SSL Inspection, a middlebox such as a firewall performs a man-in-the-middle attack on SSL connections:
 - Client sends ClientHello, which the proxy intercepts
 - Proxy does a whole SSL handshake with the server
 - The proxy completes the SSL handshake with the client, and presents a certificate for the server, that the proxy has issued. This is called a “fake certificate”.
 - The proxy decrypts and re-encrypts all traffic
 - And gets to inspect (and optionally modify) all traffic

- But wait, doesn't SSL protect against MitM ???
 - SSL Inspection only works if the proxy is an acceptable CA.
 - SSL does not protect you from rogue CAs.
 - And that is what ComodoHacker's “friends” did for connections to Gmail and others from Iran.